Knowledge-Based Systems 105 (2016) 160–174

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

DECO$_3$R: A Differential Evolution-based algorithm for generating compact Fuzzy Rule-based Classification Systems

Nikolaos L. Tsakiridis[a], John B. Theocharis[a,*], George C. Zalidis[b]

[a] Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Thessaloniki, 54124, Greece
[b] Faculty of Agriculture, Aristotle University of Thessaloniki, Thessaloniki, 54124, Greece

ARTICLE INFO

Article history:
Received 21 July 2015
Revised 6 May 2016
Accepted 8 May 2016
Available online 9 May 2016

Keywords:
Fuzzy Rule-based Classification Systems (FRBCS)
Differential Evolution
AdaBoost
Fuzzy Token Competition
Genetic Cooperative - Competitive Learning (GCCL)
Genetic Tuning

ABSTRACT

In this paper a novel Genetic Fuzzy Rule-based Classification System, named DECO$_3$R (**D**ifferential **E**volution based **Co**operative and **Co**mpeting learning of **Co**mpact F**R**BCS), is proposed. DECO$_3$R follows the genetic cooperative - competitive learning (GCCL) approach and uses Differential Evolution as its learning algorithm. In this frame, every chromosome encodes a single fuzzy rule. The proposed AdaBoost-based Fuzzy Token Competition (FTC) method is employed to deal with the cooperation - competition problem, an integral part to all GCCL algorithms. DECO$_3$R learns clear, precise and predictive rules where the fuzzy sets in the premise part are consecutive. The experimental component analysis demonstrates that DE as a learning algorithm outperforms a simple Genetic Algorithm. Additionally, the novel FTC method exceeds the performance of other similar techniques. The experimental comparative analysis highlights the robust performance of DECO$_3$R compared to other rule learning algorithms, both in terms of accuracy and of structural complexity.

1. Introduction

The most instrumental field regarding the applications of the Genetic Fuzzy Systems (GFSs) [1,2] is the development of Fuzzy Rule-based Systems (FRBSs) using Evolutionary Algorithms (EAs). FRBSs deal with linguistic IF-then type of rules, which are highly intuitive and interpretable by human beings. FRBSs pertaining to classification problems where non-fuzzy input vectors are to be assigned to one of a given set of classes, are called Fuzzy Rule-based Classification Systems (FRBCSs) [3]. The generation of FRBCSs for high dimensional datasets is challenging, considering the exponential increase of the possible fuzzy rules as the feature space increases. The incorporation of EAs (most typically of Genetic Algorithms) in FRBSs enables the rule base to be formed in a constructive and automatic way even for highly dimensional datasets, due to the their increased searching capabilities [4]. It further paved the way to improved structural complexity of the rule bases, by implementing feature selection schemes and thereby reducing the total number of features required.

Following the taxonomy proposed in [5], methods encoding a rule in a single chromosome fall within the following three subcategories (arranged in the order in which they were developed):

1. The Michigan approach [6] (also referred to as learning classifier systems), wherein the set of rules is formed through the sequential observation of the training patterns. After the formation, the rule base is tuned using an EA. UCS [7] is one of the most prominent examples of this method. The issue with this classical approach is that it does not resolve the cooperation - competition problem of the rules.
2. The Iterative Rule Learning (IRL) approach [8,9] where the algorithm iteratively learns one rule per iteration, consequently multiple runs are required in order to create a rule base. The IRL approach removes at each iteration the covered training patterns to compel new rules to focus on the uncovered patterns. Thus, at each iteration a new population of chromosomes is formed which compete among themselves so that the best chromosome is inserted into the rule set. The patterns covered adequately by the best chromosome are removed from the training set. In this approach, a mechanism must ensure that rules extracted at later stages are not conflicting with previously removed training patterns. It must be further noted that in classification problems, IRL learns one class label at a time. The drawback is that the ordering of the learned rules is not optimal as it relies on which class is learned first. Prominent

* Corresponding author. Tel.: +302310996343.
E-mail addresses: tsakirin@ece.auth.gr (N.L. Tsakiridis), theochar@eng.auth.gr (J.B. Theocharis), zalidis@agro.auth.gr (G.C. Zalidis).

http://dx.doi.org/10.1016/j.knosys.2016.05.013
0950-7051/© 2016 Elsevier B.V. All rights reserved.

examples following the IRL approach are the MOGUL [10], SLAVE [11] and FaIRLiC [12] algorithms.

3. The Genetic Cooperative Competitive Learning (GCCL) approach [13,14], in which the whole population encodes the rule base. In contrast to IRL, the rule base is not populated iteratively, but rather the whole population of chromosomes *is* the rule base. For classification problems, this allows the learning of all class labels simultaneously. Moreover, the cooperation - competition problem must be solved internally at each generation when the fitness of each chromosome is calculated. Considering that the whole set of rules is available, the rules can be re-ordered in a more optimal way as opposed to the IRL approach. A notable algorithm following the GCCL approach is GP-COACH [15], which uses Genetic Programming as its Evolutionary Algorithm, and a crisp Token Competition method to impel the rules to cooperate and compete with each other. Various other methodologies follow this approach [16–18].

In this paper, we propose a novel GFRBCS, called DECO$_3$R, utilizing Differential Evolution (DE) [19,20] as its evolutionary learning algorithm and following the GCCL approach. The motivation behind the development of DECO$_3$R was to create a FRBCS which is a) able to handle high and very high dimensional data, and b) provide both an accurate and a compact FRBCS.

To this end, we elected to use DE as the EA in the FRBCS. DE is considered to be one of the most powerful optimization algorithms for real-valued problems in current use [21]. DE exhibits fast convergence [22], the space complexity is relatively low and has relatively few control parameters [19]. Additionally it is highly customizable, which allows it to be robust. It has recently been applied in the optimization of Neuro-Fuzzy Systems [23], and for solving the joint replenishment problem [24,25]. Furthermore, DE has been successfully applied in conjunction with other algorithms, such as the Artificial Bee Colony algorithm [26], the Fruit Fly Optimization Algorithm [27], and TOPSIS and Tabu Search [28].

There are mainly two major novelties incorporated into DECO$_3$R:

(1) The introduction of the DE algorithm as the EA in a FRBCS. Customarily, FRBCSs use a Genetic Algorithm as the EA. The aforementioned assets of DE attest to its adoption as a learning algorithm for FRBCSs, and in particular when dealing with high dimensional data. In Subsection 4.1 an experimental analysis is performed which statistically demonstrates the superiority of DE over the conventional approach.

(2) The novel Fuzzy Token Competition mechanism, which enforces the competition and cooperation among the rules. It involves two discrete tasks: a) the ordering of the rules based on their individual fitness, calculated in the weighted pattern space, using the principles of the AdaBoost approach [29–31] and b) the discarding of the non significant rules.

The rest of the paper is organized as follows. Section 2 describes the Differential Evolution algorithm. In Section 3 the proposed DECO$_3$R method is presented. The experimental results are given in Section 4. Finally, concluding remarks are provided in Section 5.

2. The Differential Evolution algorithm

The Differential Evolution (DE) algorithm was originally developed by R. Storn and K. V. Price in 1995 [19]. The purpose of DE is to optimize (either minimize or maximize) a given fitness function.[1] More formally, the optimization task is to find a $\vec{x}^* = [x_1, x_2, \ldots, x_D]^T$ that minimizes a fitness function $f(\vec{x})$ with f: $\Omega \subseteq \Re^D \to \Re$ such that $f(\vec{x}^*) \le f(\vec{x})\ \forall \vec{x} \in \Omega$. When no constraints exist, it follows that $\Omega = \Re^D$. The main stages of the DE algorithm are presented in Fig. 1.

[1] Without a loss of generality, for the extent of this document we will assume that the task is to minimize the fitness function.

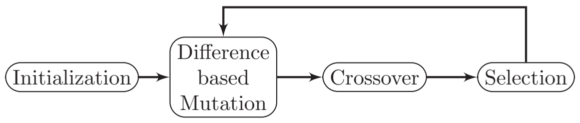

Fig. 1. The main stages of the DE algorithm.

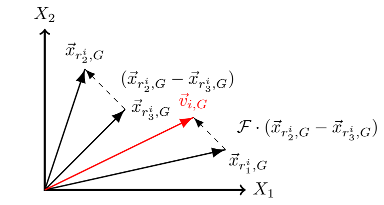
X_2
$\vec{x}_{r_2^i,G}$
$(\vec{x}_{r_2^i,G} - \vec{x}_{r_3^i,G})$
$\vec{x}_{r_3^i,G}$
$\vec{v}_{i,G}$
$\mathcal{F} \cdot (\vec{x}_{r_2^i,G} - \vec{x}_{r_3^i,G})$
$\vec{x}_{r_1^i,G}$
X_1

Fig. 2. Mutation of a chromosome using DE in $\Re^2$.

The algorithm uses a population of *NP* real-valued parameter vectors (chromosomes) with a dimension of *D* (genes). The task is to find the best solution within a G_{max} generations. The *i*-th chromosome is encoded as follows:

$$\vec{x}_{i,G} = [x_{1,i,G},\ x_{2,i,G},\ x_{3,i,G},\ \ldots\ , x_{D,i,G}] \tag{1}$$

where *G* denotes the generation index.

2.1. DE operators

2.1.1. Mutation

The novelty of the DE algorithm is that it uses the *difference* between vectors in the mutation process. At each generation, *NP* mutated vectors (also called *donor* vectors and denoted by $\vec{v}$) are created, deriving from the current population. They are created by applying a *difference vector* to a *base vector*. In its simplest form, the mutation is calculated as follows: three random and mutually exclusive indices (r_1^i, r_2^i, r_3^i) are selected, that are also different from the target vector index $i = 1, \ldots, NP$, in order to create *NP* mutated vectors. The mutation is calculated as follows:

$$\underbrace{\vec{v}_{i,G}}_{\text{donor vector}} = \underbrace{\vec{x}_{r_1^i,G}}_{\text{base vector}} + \mathcal{F} \cdot \overbrace{(\vec{x}_{r_2^i,G} - \vec{x}_{r_3^i,G})}^{\text{difference vector}} \tag{2}$$

The difference is scaled using the scalar value of $\mathcal{F}$, usually falling within the [0.4, 1] range. Greater values of $\mathcal{F}$ favor exploration of the space, since the perturbation applied is larger, while smaller ones favor exploitation, since the perturbation is smaller. A simple implementation of the mutation scheme in $\Re^2$ is presented in Fig. 2.

2.1.2. Crossover

Every donor vector $\vec{v}_{i,G}$ exchanges some of its genes with the base vector $\vec{x}_{r_1^i,G}$ in order to form the *trial* vector, denoted by $\vec{u}$. In the DE literature, two forms of crossover are usually employed: the *exponential* crossover (also known as two-point modulo crossover) and the *binomial* crossover (also known as uniform crossover) [21].

Exponential crossover. Initially a random integer *n* within [1, *D*] is selected, acting as the starting point of the operation. Afterwards, the length of the crossover operation is selected using Algorithm 1, where $\mathcal{U}(0, 1)$ denotes the uniform random distribution in the